Detecting Atmospheric Rivers in Large Climate Datasets

Surendra Byna, Prabhat, Michael F. Wehner, and Kesheng Wu

Lawrence Berkeley National Laboratory
One Cyclotron Road
Berkeley, CA 94720

Lawrence Berkeley
National Laboratory

Detecting Atmospheric Rivers in Large Climate Datasets

Surendra Byna* Prabhat† Michael F. Wehner‡ Kesheng Wu§

Lawrence Berkeley National Laboratory, Berkeley, CA

ABSTRACT

Extreme precipitation events on the western coast of North America are often traced to an unusual weather phenomenon known as atmospheric rivers. Although these storms may provide a significant fraction of the total water to the highly managed western US hydrological system, the resulting intense weather poses severe risks to the human and natural infrastructure through severe flooding and wind damage. To aid the understanding of this phenomenon, we have developed an efficient detection algorithm suitable for analyzing large amounts of data. In addition to detecting actual events in the recent observed historical record, this detection algorithm can be applied to global climate model output providing a new model validation methodology. Comparing the statistical behavior of simulated atmospheric river events in models to observations will enhance confidence in projections of future extreme storms.

Our detection algorithm is based on a thresholding condition on the total column integrated water vapor established by Ralph et al. (2004) followed by a connected component labeling procedure to group the mesh points into connected regions in space. We develop an efficient parallel implementation of the algorithm and demonstrate good weak and strong scaling. We process a 30-year simulation output on 10,000 cores in under 3 seconds.

Index Terms: J.2 [Computer Applications]: Physical Sciences and Engineering— [I.5.4]: Pattern Recognition—Applications; I.6.6 [Simulation and Modeling]: Simulation Output Analysis—

1 INTRODUCTION

Extreme precipitation events on the western coast of North America are often traced to an unusual weather phenomenon known as atmospheric rivers (ARs). These events refer to filamentary structures in atmosphere that transport significant amounts of water over a long distance in narrow bands [2, 15]. In one of the earliest studies on this phenomenon, it was determined that such a structure could carry more water than the great river Amazon [12]. Figure 1 shows an example of an atmospheric river that deposited record amounts of rainfall on California over the course of several days in December 2010. For regions such as the west coast of the United States, atmospheric rivers bring more than half of the annual total precipitation and can occur in as few as five days [2]. Their intensity creates a possibility of flooding and wind damage, yet at the same time they provide a significant amount of the fresh water needed for the western states' water management systems. Although current research is focussed on AR events making landfall on the western coast of North America, the phenomena is not limited to the northeastern Pacific and can occur in other ocean basins.

This study of atmospheric rivers is part of on-going efforts to understand the mechanisms responsible for severe but infrequent

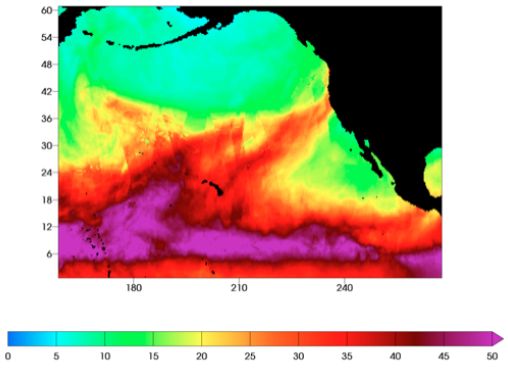

Figure 1: The observed three day average total precipitable water (mm) on December 14, 2010 from an analysis of SSM/I satellite data (www.rss.com). The long filamentary structure reaching west coast of US is known as an atmospheric river.

weather events. In some winter-time events, such as atmospheric rivers, several planetary-scale conditions must be in phase for such large entrainments of tropical moisture [15]. To reach more general conclusions, many atmospheric events must be analyzed individually and as a whole set. To analyze these events, they must first be identified.

In this work, we develop an efficient algorithm for identifying atmospheric rivers from both observational data from satellite measurements and climate model output data. Part of our motivations are to understand the statistical behavior of AR events to understand how they might change in a warmer climate. Hence, a key objective is to develop an efficient algorithm to identify AR from large volumes of data, allowing us to determine the frequency and intensity of AR events. Additional information about the structure of AR events, notably landfall location, intensity and duration are also obtainable by our method and will prove useful in projection of future climate change.

Observed precipitation and offshore wind speed [11] have been used to identify atmospheric rivers in the western Pacific basin by constructing a scatterplot of high quality hourly precipitation and wind data collected at key coastal weather stations [20]. This ad hoc method is based on setting thresholds of precipitation and wind speed in the upslope direction and has proved useful in identifying recent atmospheric river events. However, this detection method is localized by definition and requires ancillary data, such as total precipitable water from satellite measurements, to characterize the atmospheric river event. Furthermore, as atmospheric rivers can happen in any ocean basin, the scheme would fail if the event does not make landfall where quality observations are available. This likely precludes analyses in a global context. Application to climate simulations may also pose problems due to model bias in precipitation and wind fields. For example, the thresholds appropriate to observations may not work well if model extreme precipitation and winds exhibit systematic errors. In this paper, we present an alternative detection scheme based on examining basin-wide data

*e-mail: SByna@lbl.gov
†e-mail:prabhat@hpcrd.lbl.gov
‡e-mail:mfwehner@lbl.gov
§e-mail:KWu@lbl.gov

characteristic of the atmospheric river phenomenon. Our methodology allows for detection and characterization of such events in both satellite measurements and climate model output. As such, it will prove a critical tool in the projection of future changes in this class of extreme storm.

The key contributions of this work are as follows:

- We designed an efficient algorithm for identifying AR using total column integrated precipitable water vapor data from either observations or simulations. This algorithm is designed to work with water vapor data alone, without ancillary data, which makes it possible to apply the same detection algorithm anywhere on the globe.
- Our algorithm is highly parallelizable; we demonstrate efficient parallel scaling on a large 1TB dataset. We believe that our techniques can be applied to the next generate climate models which will produce petabytes of data.
- We verify the results from our algorithm against published studies by using a set of satellite data that have not been previously used for this purpose. The data used in this study is from Advanced Microwave Scanning Radiometer (AMSR-E) satellite described in Section 4. We obtain classification accuracy of 92%.

2 Related Work

In this section, we briefly review related work on atmospheric rivers, climate modeling, and feature detection algorithms.

2.1 Atmospheric Rivers

An atmospheric river is a long and narrow structure in atmosphere that transports tropical moisture to the far-flung regions outside of the tropical zone[2, 15]. Zhu and Newell were the first to name this phenomenon "atmospheric river" noting that they typically transport more water than the Amazon[23]. As they can be highly localized, river is an apt description of such a narrow stream of moisture moving at high speeds across thousands of kilometers. AR such as the one shown in Figure 1 passes near the Hawaiian Islands are often called the Pineapple Express by television weather reporters. AR events occur in oceans around the globe, including the Atlantic basin affecting the British isles. [1].

The key characteristics recognized in earlier studies of ARs is the moisture flux [24]. However, that quantity turns out to be a hard to directly observe. In 2004, Ralph et al. [16] established a much simpler set of conditions for identify atmospheric rivers in satellite observations. Their detection works with two-dimensional data over a uniform mesh on the global and is primarily based on the Integrated Water Vapor (IWV) content, which measures the total water content (measured in volume) in the volume of atmosphere above a unit of earth surface. This quantity is measured in millimeters (mm) or centimeters (cm). More specifically, they identify atmospheric rivers as atmospheric features with IWV $>$ 2cm, more than 2000 km in length and less than 1000 km in width. Based on this definition, Ralph and colleagues have identified hundreds of atmospheric river events in the data produced by Special Sensor Microwave Imager (SSM/I) satellite observations [2, 10].

In this work, we will use a different set of satellite data as well as output data from a state of the art high-resolution climate model. The observational data we use is from a satellite called Advanced Microwave Scanning Radiometer (AMSR-R). This device measures IWV allowing us to use the same conditions as proposed by Ralph et al. [16].

Objective identification of atmospheric river events is a challenging task. Identifying observed events in the historical record for case study analyses can exploit associated information such as onshore extreme precipitation and wind direction to identify candidate events. Large scale structural information can then be gained by analyses of satellite measurements [15]. However, analyses of the statistical behavior of atmospheric rivers are also necessary to understand the more general relationship to large-scale climatic variations. The ability of climate models to simulate atmospheric river statistics is key to projecting if these phenomena change as the climate warms. Hence, an atmospheric river identification scheme that neither misidentifies nor misses candidate events is critical to the statistical analysis of climate models, and their comparison to the observed recent past.

2.2 Climate Models

Using computers to simulate and predict the weather is one of the most successful applications of computer technology in the past few decades. With this technology, we are able forecast severe weather and effectively evacuate the affected area. The same technology has also been used to assess how the climate changes will impact human society in the future [14].

The Community Earth System Model (CESM) [2] is a fully-coupled, global climate model developed by the National Center for Atmospheric Research (NCAR) in Boulder, Colorado [3]. It is one of the widely used global climate modeling tool used for climate research. In our study of atmospheric rivers, we will be using the output from a component of CESM called fvCAM, the finite-volume version of the Community Atmosphere Model.

Data produced by climate models and various satellites orbiting the earth is massive. For example, 15-year modeling data from fvCAM with output every 6 simulated hours, and a mesh point resolution of 0.5° latitude by 0.625° longitude produces roughly 500GB of data [19]. Ensembles of simulations can easily produce petabytes of data. The Advanced Microwave Scanning Radiometer (AMSR-E) satellite produces 150GB data for a decade of observations. There are many satellites scanning the earth for different observations. Together they produce many terabytes of data.

In this work, a key goal is to develop an algorithm for identifying atmospheric rivers from such datasets. To work with petabytes of data, our algorithm has to be efficient and able to take advantage of massively parallel computers.

2.3 Feature Detection on Mesh Data

Climate Model and satellite output are typically generated (or regrided) on a regular mesh over the globe. Following the methodology used by Ralph et al., we perform our detection on 2-D data on the latitude-longitude mesh [16]. An atmospheric river is an event that can last for a few days. Our detection algorithm processes one day at a time. For each day's data, the AR appears as a connected regions in space where the integrated water vapor content is high. This type of feature in space is commonly known as *region of interest*. Identifying such regions of interest is a basic operation in many computer vision and visual analysis tasks [13, 6].

Our detection algorithm proceeds in three steps. The first step performs a thresholding operation based on IWV value; mesh points with high IWV values are marked for further processing. The second step connects the marked mesh points into regions. This step employs a connected component labeling algorithm. The connected regions are passed to the last step for verification of sizes. The first and last steps are relatively straightforward and can be found in many textbooks [6, 13]. In this section, we briefly review the algorithms used for connected component labeling [4, 21].

[1] http://cimss.ssec.wisc.edu/goes/blog/archives/3838.

[2] http://www.cesm.ucar.edu/.

[3] http://ncar.ucar.edu/.

The IWV data processed by our feature identification procedure is stored as a 2-dimensional array. The output from the thresholding step can be treated as a binary image, where the foreground pixels are mesh points with large IWV values and the background pixels are mesh points with small IWV values. This allows us to use the connected component labeling algorithms developed from image processing. There are a variety of algorithms for this task. For example, there are a number of different parallel approaches [7, 18], some methods using specialized hardware [5, 9], and some using GPU-type accelerators [8]. In our application, the image sizes are relatively modest. For example, at 0.5° resolution, the whole global is divided into a mesh of 720×360, which has just about a quarter of a million mesh points (or pixels in the binary image). Due to this modest size, we choose to perform connected component labeling using only a single CPU core.

The sequential labeling algorithms can be divided into three categories based on how many times the binary image is accessed [17, 21]. The simplest algorithms requires multiple passes through the binary image and are known as multi-pass algorithms. The algorithm by Suzuki et al. [17] is an efficient example in this category. The second category of labeling method is known as two-pass algorithms because they require two passes through the binary image, once to gather the connectivity information among the foreground pixels and once to assign the final labels to each pixel. The third category of labeling methods requires only a single pass through the data [1]. On modern CPUs, memory accesses typically dominate the cost of the connected component labeling algorithms. In this case, the algorithm that scan through the data array the least number of times should be the fastest. This argument favors the one-pass algorithms. However, the one-pass algorithms have to perform random accesses, which are typically much slower than sequential memory accesses. For this reason, an efficient two-pass algorithm can actually outperform the best of the one-pass algorithms [21]. For this reason, we choose to use a two-pass algorithm in this work.

The two-pass algorithms need some data structures to record the equivalence information among the provisional labels assigned to the foreground pixels during the first pass. They avoid scanning the image multiple times by manipulating the label equivalence information to arrive at a final assignment for each provisional label. The most efficient data structure for keeping track of the label equivalence information is called union-find [3], and the most efficient implementation of the union-find data structure is an implicit data structure that uses a single array [21]. An efficient union-find implementation is critical to the overall effectiveness of the two-pass algorithm.

It is possible to represent the binary image differently to achieve better performance for the connected component labeling step [22]. The key idea is to make the representation of the foreground pixels more compact. However, representing the foreground pixels more compactly will make it more difficult to compute lengths and widths of the connected regions, which make the third step more expensive. In this work, we chose to keep the binary image in an two-dimensional array.

3 Our Approach

Our algorithm processes 2-D meshes defined over the globe. These meshes are relatively small, for example, the satellite observation data is defined on a $1/4^\circ$ mesh with just over 1M mesh points, and the climate model output uses a $1/2^\circ$ mesh. Even with fine meshes at $1/10^\circ$ mesh, the data associated with a single variable, i.e. integrated water vapor (IWV), can easily fit into main memory. While we need to process many timesteps in the complete dataset, this can be done in parallel.

A schematic illustration of the parallel algorithm for AR detection is shown in Figure 2. The algorithm can be divided into an

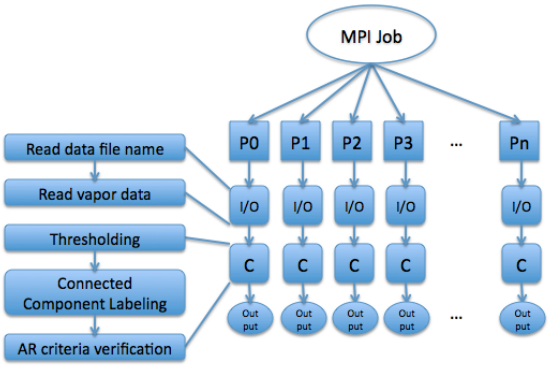

Figure 2: Schematics of AR detection tool implemented with MPI

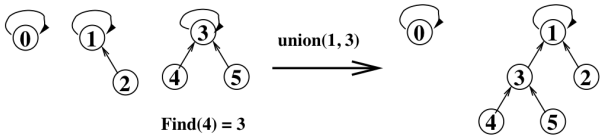

Figure 3: An array representation of the rooted trees.

I/O phase and and a Compute phase (shown as C in Figure 2). The I/O phase includes reading the input filenames and vapor data. The computation phase consists of thresholding, connected component labeling, and verification steps. Each process generates an output indicating the presence or absence of an AR. Our design allows each process to run independently without any need for inter-process synchronization or communication.

3.1 I/O Phase

Our current implementation requires a list of data file names to process. This list is currently stored in a single, shared file. Currently, all processes read the file; it is possible to split this file in the future to reduce metadata overhead, but for now we have decided to use this simple approach. Once each process determines what file to process, it then proceeds with reading the IWV data.

The function that performs the reading of IWV data takes a number of optional input parameters, such as granularity of climate data, type of data format (such as gunzip compressed format, netCDF, etc.), the number of timesteps present in one day's data and regions where AR should be detected. This flexibility allows us to detect AR in any region of the world at different granularity.

3.2 Compute Phase

3.2.1 Thresholding

In the compute phase, the first step is a set of thresholding operations on IWV values. Ralph et al. [16] specify IWV values greater than 20mm for detecting atmospheric rivers. We use this threshold value for all results reported in the paper. However, our detection tool can take on a pair of thresholds that define the a lower bound and a upper bound for the IWV values. This additional flexibility can be useful for with systematic biases. The output of the thresholding step is a collection of mesh points that satisfy the threshold criteria. These "foreground" pixels are then processed by the Connected Component Labeling (CCL) step.

3.2.2 Connected Component Labeling

Our connected component labeling implementation is based on a two-pass algorithm [21]. The algorithm can be broken down into

three steps. The first step assigns a provisional label to each mesh point visited. These provisional labels may turn out to be assigned to connected mesh points. We say that these labels are equivalent. This label equivalence information is recorded in a data structure called union-find. The second step works with the union-find data structure to determine the final label for each provision label. The third step replaces the provisional labels with their final values. This third step is a series of straightforward assignments.

The first step examines each mesh point in turn. A mesh point failing the thresholding conditions will receive a special label, say 0, to indicate that it is not of interest. A mesh point satisfying the thresholding conditions will receive a provisional label. This assignment proceeds as follows. If there is no neighbor with a provisional label already, then this mesh point receives a new label. If any of its neighbors have already received a label, any of their labels can be assigned to the current mesh point. Because the neighbors are connected to this mesh point and to each other, their labels should be the same. We say that these labels are equivalent, and choose the smallest labels as the "representative" of the groups of equivalent labels.

The union-find data structure stores the label equivalence information. This data structure supports two key operations called union and find. Given any provisional label, the find operation locates its "representative." Given any two provisional labels, the union operation is to record that they are equivalent to each other. This operation can be implemented as two find operations followed by an operation to set one "representative" pointing to the other. We choose to have the representative with larger numerical value pointing to the representative with smaller value. The union-find data structure can be interpreted as representing a forest of union-find trees, where the "representative" is the root of each tree. Pictorially, this is illustrated in Figure 3. By choosing to use non-negative integers as labels, it is possible to use the labels as the array index and implement the union-find data structure in a single array as illustrated in Figure 3.

Using an array to implicitly represent the union-find trees has the advantage that the memory for the union-find data structure is consecutive in memory. Furthermore, the find operations always traverse to the left in Figure 3. This predictable pattern reduces the average cost of the memory accesses, which improves the overall effectiveness of the labeling algorithm.

3.2.3 Verification

After the connected component labeling step, each connected group of mesh points receives a unique label for identification. We then compute the length and width of each group, and impose the relevant constraints (i.e. $Length > 2000km$ and $Width < 1000km$ [16]) in the verification step.

Our implementation can also impose additional spatial constraints. For example, to declare an atmospheric river as a "pineapple express", we can test for the AR having passed through the islands of Hawaii and the west coast of US.

4 Experimental Methodology

Thus far, we have described the algorithm for detecting atmospheric rivers. We are interested in evaluating the performance of our algorithm along the following metrics:

- How well does our algorithm perform? What is its accuracy?
- How well does the implementation scale with large data (weak scaling)?
- How well does the implementation scale with number of processes (strong scaling)?

We now describe our experimental methodology for addressing these questions.

4.1 Accuracy of our approach

We have taken a number of approaches to validate the accuracy of our detection algorithm. First, we made sure that our algorithm can accept the parameters established by Ralph et al. [16]. Theoretically, this should ensure we have the same detection algorithm as used by Ralph and colleagues.

To start off, we verified that we were able to detect recent "pineapple express" events in the recent news such the one around December 14, 2010 shown in Figure 1. This approach is useful but limited in that we only have established reports for a handful of such events. We considered mining weather reports for more occurrences of this pattern; however, we are not able to find a reliable mining tool to process such information.

We also used a subjective approach, wherein we had a domain science expert examine a number of "hard" cases and report if the algorithm was computing the right result. While this exercise was extremely insightful in terms of understanding the phenomena, we could not be absolutely certain that we had captured all occurrences. Needless to say, it was impossible to have the domain expert sift through all of the observational data due to time constraints.

We briefly considered the option of using a system like the Amazon Mechanical Turk[4] for obtaining human annotated data, but there is a certain degree of domain knowledge required to successfully complete this task.

Finally, we settled on comparing our results to the published AR events in the west coast US by a number of other researchers [2, 10]. These papers contain an exhaustive list of atmospheric rivers reaching the US west coast from the year 1998-2008. We treat the results reported in Dettinger et al. from June 2002 and 2008 as ground truth. We note that our results are obtained from a different satellite, Advanced Microwave Scanning Radiometer (AMSR-E) satellite [5].

4.2 Weak Scaling

The field of climate modeling is undergoing active research: we expect larger and larger simulation datasets to be produced in the coming years. While the dataset sizes are increasing, we also have access to large supercomputing systems to process the data. Hence, it is important that data analysis programs are able to scale up as more computing resources are provide for large data sets. To measure this type of scalability, we keep the work given to each process constant, but increasing the number of processes across 1000, 2000, 4000, 8000, and 10000 MPI processes, while proportionally increasing the problem sizes from 50GB to 1TB.

In our tests, we will report both the time to read the input data and the time to complete the computations. In measuring the I/O performance, we will report the I/O throughput instead of the more common read or write speed. There is no synchronization among the processes, therefore, the I/O operations on each process are not coordinated. In this case, we measure the average I/O performance on each process separately and report the sum of the I/O performance as the aggregate I/O throughput.

4.3 Strong Scaling

The strong scaling refers to the ability of an algorithm to take advantage of more computing resources to complete the same task. In our case, we keep the input data size fixed at 1TB and increasing the number of processes from 100, 200, 500, 1,000, 2,000, 5,000 to 10,000 MPI processes. This data set has 10,000 days of global climate modeling data; therefore we test scaling upto 10,000 processes.

[4] `http://aws.amazon.com/mturk/`.
[5] Web URL is `http://www.ssmi.com/`

4.4 Hardware Platform

We conducted our experiments on the NERSC Cray XE6 supercomputing system Hopper [6]. The system has ≈6,400 compute nodes, with 24 cores (total ≈150,000 cores, 2 twelve-core AMD 'MagnyCours' 2.1 GHz processors per node) and 32GB memory per node. We used all 24 cores of a node for our tests and have one MPI process on each core. Hopper uses Lustre as its file system, with a peak theoretical I/O bandwidth of 35GB/s. The parallel file system is configured with 156 Object Storage Targets (OSTs).

4.5 Data

4.5.1 Observational Data

We use a geophysical dataset derived from observations collected by the AMSR-E satellite. The overall dataset contains sea surface temperature, surface wind speed, atmospheric water vapor, cloud liquid water, and rainfall rate. The orbital data of the satellite is mapped to 0.25° mesh, i.e. each of the data observations is gridded onto a 1440 x 720 matrix. The daily data collected by AMSR-E contains gaps because the satellite can not cover the whole globe in a day. To obtain complete data for any given day, RSS provides time-averaged data using a 3-day moving window.

In our atmospheric river detection scheme, we use the vertically integrated water vapor data from files containing 3-day averages of column integrated water vapor. The files are compressed into gzipped format (.gz). We converted this compressed files into netCDF format. The size of each 3-day average file in netCDF format is 40 MB. In our tests, we used observation data for 3100 days, which amount to 124 GB. This dataset is used for verifying the accuracy of our tool in detecting atmospheric rivers in the coastal areas of California, Oregon, and Washington states. We compare the results with the manually identified list of AR events by Dettinger et al [2].

4.5.2 Model Data

We use climate data generated by the finite volume version of the Community Atmospheric Model (fvCAM) in our scalability study [19]. The fvCAM uses a finite volume approximation to the atmospheric equations of motion and have been specifically optimized for parallel execution. Output data in netCDF format includes multiple variables such as pressure, humidity, temperature, total vertically integrated water vapor. For detecting atmospheric rivers, we use the data value for the integrated water vapor. The data is arranged in a 361 x 576 mesh, which represents 0.5° latitude by 0.625° longitude. There are 4 simulated time steps per one day, i.e. one per every 6 hours, and the total dataset contains 15 simulated years worth of data that amounts nearly 450 GB. The dataset is stored into 1095 files and each file consists of 5 days worth of data.

To avoid dealing with intra-day variations, our detection algorithm works with daily averages calculate from the 6-hour timesteps within the day. Since model data does not have any missing data, we did not need to compute the average for 3 days as in the observational data. In our strong scaling tests, we used data related to 10,000 days, which is ≈1 TB. In the weak scaling experiments, the data size is increased in proportion with the number of processes used. In these, each MPI process analyzes data related to one day. For example, in a 10,000 process MPI job, the application processes 10,000 days worth of data, which is in the range of 1 TB. Similar to the observational data analysis, in both weak scaling and strong scaling studies, we analyzed data related coastal areas of California, Oregon, and Washington states. The tool can be used for detecting AR in any region, by changing the longitude and latitude bounding box parameters and the AR detection criteria.

[6] http://www.nersc.gov/nusers/systems/hopper2/

Figure 5: Yearly statistics of Atmospheric river events from observational data from http://www.remss.com/amsre/.

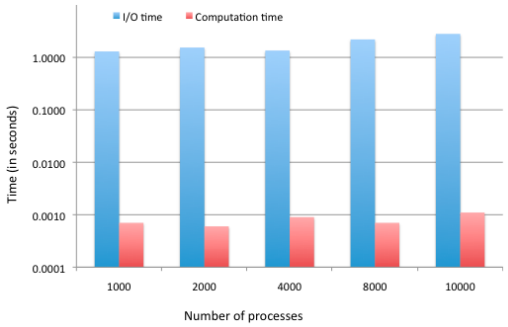

Figure 6: Weak Scaling times.

5 Results

We outlined three questions to address in our performance study. In this section, we report our findings on each separately.

5.1 Classifier Performance

We applied our AR detection tool to the observational data, and compare the detected events with the published paper by Dettinger et al [2]. We use the same thresholds listed in their paper: water vapor (>20mm), length (>2000km) and width (<1000km), and spatial constraints of examining ARs originating in the tropics and making landfall on the western US coast. Figure 4 shows a sampling of detections from our program.

Our tool is able to detect 81% of the AR events reported in Dettinger, et al. Upon further examination, we discovered that Dettinger, et al. were reporting ARs even if the river did not actually make landfall (but was close to it). Our algorithm declares an AR only if the river touches the land. We thereafter removed entries from Dettinger, et al.'s list that did not make landfall. The resulting accuracy of our tool is 92%. The remaining undetected events have vapor below threshold in some parts of the narrow band, which CCL algorithm counts as different connected components. Since these disconnected labels do not fit in the source, destination, and length criteria, they are not detected as AR by the tool. We will address this issue in future refinements of our implementation.

Figure 5 shows statistics of AR events between 2002 and 2010. For year 2002, the data is available from June to December, and for all other years, the events are for the whole year. We counted consecutive days with an AR as one event. We separate the AR events in the winter-time from summer months. This relative distribution is quite similar to those reported in earlier studies [2, 10].

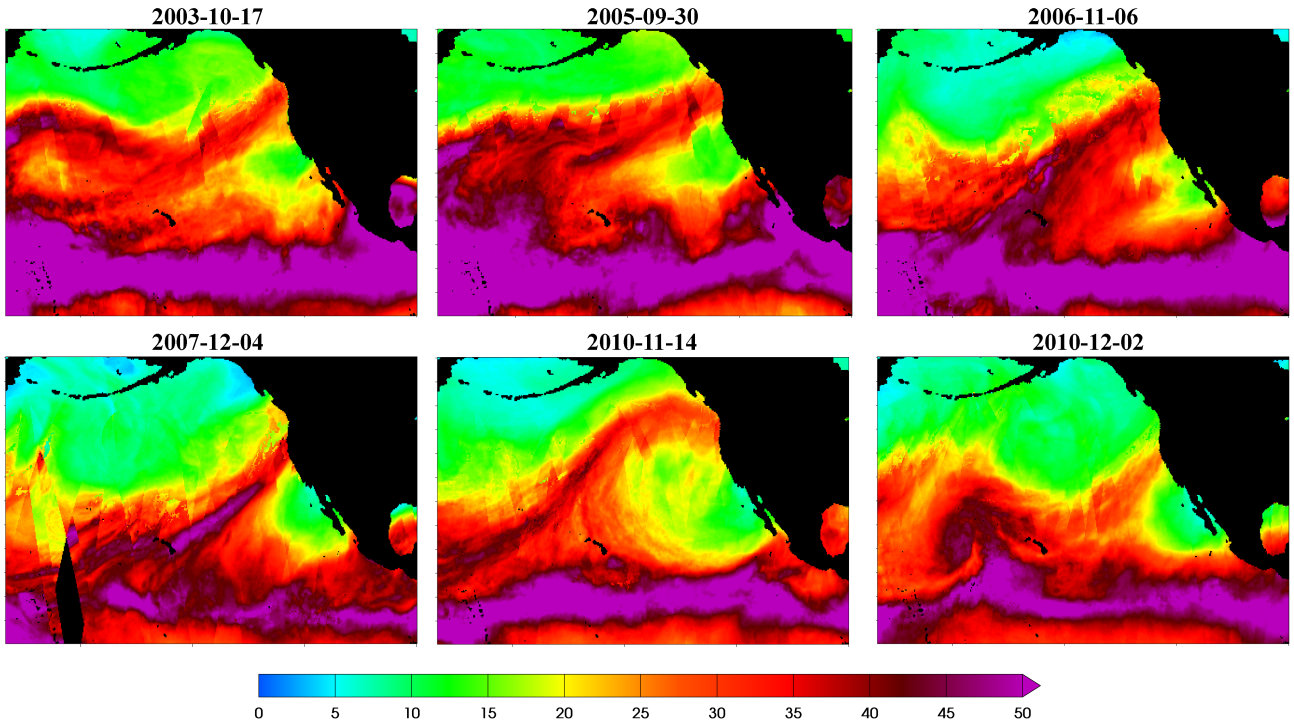

Figure 4: Some typical atmospheric river events detected by our new method from the observational dataset. Shown is total column integrated precipitable water in mm. Note that the structure of each event is unique. Also note that data irregularities in the satellite measurements (seen as abrupt discontinuities e.g. in the 2007-12-04 event) do not have an adverse effect on the detection procedure.

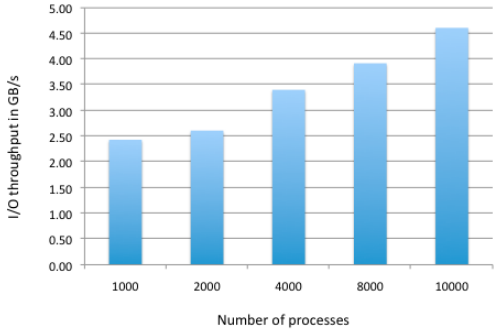

Figure 7: I/O performance for weak scaling.

5.2 Weak Scaling

Figure 6 shows results from our weak scaling experiment. The x-axis shows number of MPI processes, and the y-axis shows the time in seconds (in logarithmic scale). To recall the experimental setup, each process analyzes data for a single day; as more processes as added, the detection algorithm works on a proportionally larger number of days.

We observe that the majority of the execution time of our tool is dominated by I/O (98%). Since each process only works on one day's worth of data, we expect the the I/O time and the computation time to remain constant as the number of processes increases. While these costs stayed relatively constant, we noticed a small increase in the observed time. We attribute this to a small fraction of MPI processes taking longer than others to finish their processing. As the number of processes increase from 1000 to 10000 processes, the observed computation time fluctuates between 0.7ms and 1.1ms, we believe that these fluctuations are random in nature (and not systematic).

The I/O time also increases slightly; the main reason for this increase is due to shared access to the same input file for reading filenames. As the number of processes increase, the time to read the file names increase from 0.29s for 1000 processes to 1.54s for 10,000 processes. In the same tests, the time to read the integrated water vapor data remains about the same, 1.01s 1000 processes and 1.32s for 10,000 processes.

In Figure 7, we show the aggregate I/O throughput against the number of processes. While the previous figure (Figure 6) shows I/O time for the process that took the longest time, this figure shows aggregate I/O bandwidth of all processes. We calculated the aggregate I/O throughput as the sum of I/O throughput at each process. Since each process runs independently without any synchronization, measuring global I/O bandwidth for the application does not reflect I/O performance of the tool. As the number of processes increases, the I/O throughput also increases. In the model dataset each file contains five days of data and is therefore shared by five processes. Since each read request fetches a full stripe of data (1 MB in Lustre file system configured on Hopper), this 1 MB stripe of data includes all water vapor data for all five days. This explains achieving good I/O performance even though each process is expected to read only about 128 KB of data.

Another reason for the increased I/O throughput is because as more processes are making I/O requests to the file system, they collectively occupy more attention of the file system. Therefore, the observed throughput is a larger fraction of the file system's peak performance. In our tests, we see the aggregate I/O throughput increase from 2.5 GB/s with 1000 processes to about 4.6 GB/s with 10,000 processes.

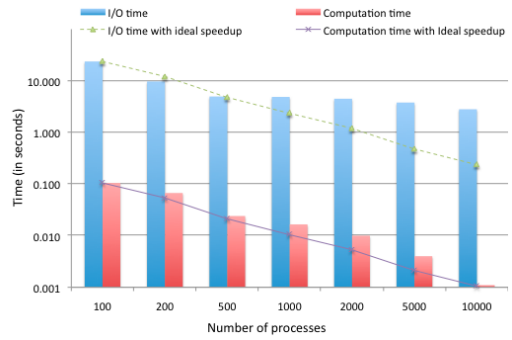

Figure 8: Strong Scaling times.

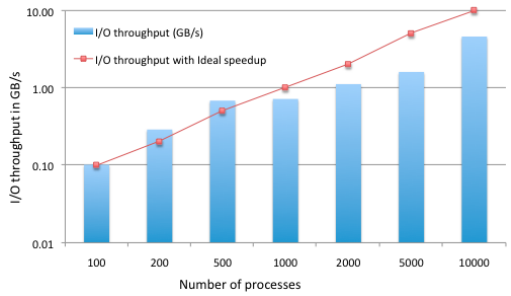

Figure 9: I/O performance for strong scaling.

5.3 Strong Scaling

Figure 8 shows strong scaling results with a fixed data size related to 10,000 days. This experiment shows how the application scales as the number of processes increase, while the data size is fixed. The two bars show the I/O time and the computation time. The sum of these two costs is equal to the total execution time of the algorithm. The upper trend line (dashed) refers to the I/O time if ideal speedup were achieved and the lower trend line represents computation time if ideal speedup were achieved. We calculated the time with ideal speedup in reference to the measured time when 100 processes were used. For instance, if the I/O overhead with 100 processes is *t*, then the I/O overhead with 200 processes is *t/2* and that with 500 processes is *t/5*, and so on. The combination of reading file names from input file and the reading vapor data from climate data set dominate the overall execution time. The total I/O time constitutes 99% of the execution time.

In Figure 8, we see that the computation time speedup generally agrees with ideal scaling. This suggests that the computations are relatively load-balanced and amenable to parallelization. In this case, each process handles data from a number of different days, which minimizes the effect of random fluctuations discussed earlier.

The I/O times are very close to ideal speedup for the test cases with 100, 200 and 500 processes. As indicated before, five processes read from a single data file and their read operations are most likely served by a single disk read, which means that 100 OSTs can serve 500 processes. In going from 100 to 500 processes, our program is effectively using more OSTs from the file system, therefore the I/O time scales well. As more processes are used, it is no longer possible to have each OST serve five processes. This creates I/O contention and increases the time needed to complete the I/O operations. We see that the I/O time in Figure 8 goes above the expected value for ideal speedup when more than 1000 processes are used.

In Figure 9, we show the aggregate I/O throughput as the number of processes increases with fixed data size. The I/O throughput increases as the number of processes increase up to 500 processes following the linear trend. From the 1000 processes case, the I/O throughput falls short of ideal growth. Nevertheless, the aggregate throughput still increases, reaching 4.6 GB/s with 10,000 processes.

Our results indicate that our tool is quite well suited for weak scaling. Our tool can analyze more data with a larger number of processes. This will be useful in processing output data from century scale climate model integrations.

6 Future Work

While our current implementation has produced very promising results, there are number of issues that we would like to address in future work. As indicated earlier, I/O accounts for the majority of processing time in our tool, we would like to undertake a more careful study of I/O optimization strategies and incorporate them into our tool. This includes avoiding the use of a single shared file-list, and more optimal strategies for multi-core read operations.

In terms of correctness, we have treated the manual results from Dettinger, et al. as ground truth. First, we would like to more closely examine results which are flagged as ARs by their approach, and missed by our implementation. We envision using more sophisticated image processing techniques (e.g. Bilateral Filtering) or topological techniques, which will make our technique more robust to choices of thresholds. We would also like to verify the detection results from our tool on climate modeling output. It is quite possible that we will need to automatically adjust threshold values for integrated water vapor for simulation output. Not all climate models produce integrated water vapor as part of their output, and we need to explore a generic way of computing integrated water vapor in model data.

7 Conclusions

Atmospheric rivers are a type of rare weather event capable of transporting large amounts of water from tropical region to elsewhere. They are a important source of fresh water as well as a cause of severe flooding and wind damage. In this work, we have developed an efficient detection tool for automatically detecting ARs. We use a combination of thresholding, connected component labeling and verification steps to check for the presence of ARs. Our implementation was able to successfully detect 92% of ARs that make landfall; the results were verified against a manually curated results published by Dettinger, et al. We demonstrated good weak and strong scaling for our implementation. We applied our tool to a large 1TB dataset on 10,000 cores, and completed the processing in 3 seconds. We believe that our fully automated and highly parallelizable tool will enable climate scientists to effectively tackle large data challenges from next generation climate simulation output.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy (DOE) by the Lawrence Berkeley National Laboratory (LBNL) under contract DE-AC03-76SF00098 (LBNL) and with support from the Office of Science (BER), U. S. Department of Energy. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] F. Chang, C.-J. Chen, and C.-J. Lu. A linear-time component-labeling algorithm using contour tracing technique. *Comput. Vis. Image Underst.*, 93(2):206–220, 2004.

[2] M. D. Dettinger, F. M. Ralph, T. Das, P. J. Neiman, and D. R. Cayan. Atmospheric rivers, floods and the water resources of california. *Water*, 3(2):445–478, 2011.

[3] L. di Stefano and A. Bulgarelli. A simple and efficient connected components labeling algorithm. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 322, Washington, DC, USA, 1999. IEEE Computer Society.

[4] M. B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 39(2):253–280, 1992.

[5] H. Flatt, S. Blume, S. Hesselbarth, T. Schünemann, and P. Pirsch. A parallel hardware architecture for connected component labeling based on fast label merging. In *ASAP*, pages 144–149. IEEE Computer Society, 2008.

[6] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, New Jerse, 2nd edition, 2002.

[7] J. Greiner. A comparison of parallel algorithms for connected components. In *SPAA '94: Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures*, pages 16–25, New York, NY, USA, 1994. ACM.

[8] K. Hawick, A. Leist, and D. Playne. Parallel graph component labelling with gpus and cuda. *Parallel Computing*, 36(12):655–678, 2010.

[9] C.-Y. Lin, S.-Y. Li, and T.-H. Tsai. A scalable parallel hardware architecture for connected component labeling. In *ICIP*, pages 3753–3756. IEEE, 2010.

[10] P. J. Neiman, F. M. Ralph, G. A. Wick, Y.-H. Kuo, T.-K. Wee, Z. Ma, G. H. Taylor, and M. D. Dettinger. Diagnosis of an intense atmospheric river impacting the pacific northwest: Storm summary and offshore vertical structure observed with COSMIC satellite retrievals. *Monthly Weather Review*, 136(11):4398–4420, 2008.

[11] P. J. Neiman, A. B. White, F. M. Ralph, D. J. Gottas, and S. I. Gutman. A water vapour flux tool for precipitation forecasting. *Proc. Institution of Civil Engineers – Water Management*, 162:83–94, 2009.

[12] R. E. Newell, N. E. Newell, Y. Zhu, and C. Scott. Tropospheric rivers? – A pilot study. *Geophysical Research Letters*, 19(24):2401–2404, 1992.

[13] M. Nixon and A. S. Aguado. *Feature Extraction in Computer Vision and Image Processing*. Newnes, Oxford, UK, 2002.

[14] R. Pachauri and A. Reisinger, editors. *Climate Change 2007: Synthesis Report – Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*.

[15] F. M. Ralph, P. J. Neiman, G. N. Kiladis, K. Weickmann, and D. W. Reynolds. A multiscale observational case study of a pacific atmospheric river exhibiting tropical-extratropical connections and a mesoscale frontal wave. *Monthly Weather Review*, 139(4):1169–1189, 2011.

[16] F. M. Ralph, P. J. Neiman, and G. A. Wick. Satellite and CALJET aircraft observations of atmospheric rivers over the eastern north pacific ocean during the winter of 1997/98. *Monthly Weather Review*, 132(7):1721–1745, 2004.

[17] K. Suzuki, I. Horiba, and N. Sugie. Linear-time connected-component labeling based on sequential local operations. *Comput. Vis. Image Underst.*, 89(1):1–23, 2003.

[18] K.-B. Wang, T.-L. Chia, Z. Chen, and D.-C. Lou. Parallel execution of a connected component labeling operation on a linear array architecture. *Journal of Information Science And Engineering*, 19:353–370, 2003.

[19] M. F. Wehner, G. Bala, P. Duffy, A. A. Mirin, and R. Romano. Towards direct simulation of future tropical cyclone statistics in a high-resolution global atmospheric model. In *Advances in Meteorology*, volume 2010, page 915303, 2010.

[20] A. B. White, F. M. Ralph, P. J. Neiman, D. J. Gottas, and S. I. Gutman. The NOAA coastal atmospheric river observatory. In *34th Conference on Radar Meteorology*.

[21] K. Wu, E. Otoo, and K. Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Analysis & Applications*, 12(2):117–135, 2009.

[22] K. Wu, R. R. Sinha, C. Jones, S. Ethier, S. Klasky, K.-L. Ma, A. Shoshani, and M. Winslett. Finding regions of interest on toroidal meshes. *Computational Science & Discovery*, 4(1):015003, 2011.

[23] Y. Zhu and R. E. Newell. Atmospheric rivers and bombs. *Geophysical Research Letters*, 21(18):1999–2002, 1994.

[24] Y. Zhu and R. E. Newell. A proposed algorithm for moisture fluxes from atmospheric rivers. *Monthly Weather Review - USA*, 126(3):725–735, 1998.